Space Station Platform Management System (PMS) Replanning Using Resource Envelopes

by
Joy Lee Bush, Anna Critchfield, and Audrey Loomis
Computer Sciences Corporation, System Sciences Division

Abstract

One of the responsibilities of the Space Station Platform Management System (PMS) is to maintain constraint-free, short-term plans for platform and free-flyer activities. Both the replanning function and the associated constraint-checking function are viewed as potentially requiring expert system assistance.

This paper describes the PMS Resource Envelope Scheduling System (PRESS) expert system, which is currently under development. PRESS capabilities will include the following:

- o Plan, replan, and perform constraint checking using resource envelopes resembling those required for telescience
- o Initialize itself using the results of a previous run
- o Infer the replanning needs associated with a change in resource availability
- o Allow the user to determine the level of interaction (including an advisory capability) with the system during execution.
- o Generate both a graphic timeline and a report as output

PRESS is being developed on an IBM PC/AT using TeKnowledge, Inc.'s M.1 expert system shell. PRESS activity definitions and constraints are based on those defined for the Cosmic Background Explorer (COBE) mission scheduled for launch in early 1989.

I. Background

The development of the Platform Management System (PMS) Resource Envelope Scheduling System (PRESS) is part of an on-going task contracted to Computer Sciences Corporation by the Mission Operations Division (code 511) and the Data Systems Technology Division (code 520) of NASA at Goddard Space Flight Center. PRESS is a prototype expert system being developed as part of an effort to study the feasibility of using expert system technology in the Space Station environment. Thus PRESS attempts to implement some of the functions that have been defined for the PMS, and to use the "resource envelope" concept that has been developed for Space Station applications but is not yet fully defined.

II. PMS Functionality

The PMS (as defined in the PMS Definition Document, October 1986) is a software system that provides operational management services among payloads and platform systems. It consists of an automated on board segment, the Platform Management

Application (PMA), and a ground segment, the Platform Management Ground Application (PMGA). Seven functions have been defined for the PMS:

(1) Short-Term Plan Management - modify, as necessary, a short-term plan in response to requests from operators, customers, subsystems and payloads;
(2) Schedule Execution - execute the short term plan by coordinating instructions to payloads and systems;
(3) Operations Monitoring and Activity Logging - track and store data for anomaly investigation and billing;
(4) Intersystem/Payload Testing - execute testing as desired by platform operators;
(5) Conflict Recognition and Resolution - recognize and prevent conflicting activities;
(6) Fault Handling - supervise fault management and reconfiguration for payloads and systems; and
(7) Transaction Checking - control messages to on board destinations.

III. PRESS as a Subset of PMS Functionality

The goals of PRESS are: to research the feasibility of expert system applications in providing PMS functionality; to use (and therefore help define) the resource envelope concept as a basis for automatic scheduling; to implement COBE functions as a proof-of-concept; and to evaluate the suitability of the system development environment for expansion of this prototype or more complex development/delivery efforts.

PRESS addresses two of the PMS functions. The first of these is the Short-Term Plan Management function. This function involves the PMGA receiving a plan from the Platform Support Center, and uplinking appropriate portions to the PMA. The PMGA and the PMA may receive plan changes requested by operators, customers, subsystems, and payloads. The second function, Conflict Recognition and Resolution, involves the monitoring of resource usage, allocation, and margins. Conflicts for resource usage are to be resolved on a priority basis. This function will be used in deciding whether a given request may be scheduled. The PMA and the PMGA are required to modify the short-term schedule while maintaining a conflict-free plan that does not exceed the platform's resource capabilities or compromise its safety.

Another PMS issue that will be explored via PRESS is how autonomous the scheduling/rescheduling functions can be made for eventual on board usage. The process of PRESS development will help to identify specific areas where human intervention is critical. Functions that can be safely migrated to the on board system will be identified.

PRESS will implement the short-term plan management function and the conflict recognition and resolution function using specific constraints from the Cosmic Background Explorer (COBE) spacecraft. PRESS will perform both initial scheduling and replanning to accommodate updates; will constantly update resource usage with each schedule modification; and will check constraints and limitations such as safety margins via a table lookup. PRESS is designed with provision for as much flexibility as possible, so that the demonstration of the rapid prototype can elicit feedback regarding which approaches appear the most useful. The prototype under development involves a smaller set of resources and constraints (primarily uplink and downlink time slots) than will be encompassed by the PMS, but the application will demonstrate basic functionality in

serving as a proof-of-concept for using the expert system approach in PMS functions. Figure 1 depicts the placement of an expert system that would include the PRESS functions in the PMS.

IV. PRESS Development Status

A final PRESS prototype system is planned for September 1987, with a rapid prototype demonstration scheduled for early May 1987. Table 1 lists the functions of the final prototype and indicates whether they will be implemented in the rapid prototype. Briefly, the rapid prototype will create a new schedule, and will accept scheduling requests either from a file or manual input. The rapid prototype expects some operator interaction and provides the advisory capability to suggest alternative time slots. It generates output in the form of a graphic timeline and a printed report. Feedback following the rapid prototype demonstration will be critical in determining the direction of further PRESS development. Section VI discusses PRESS functionality in detail.

V. PRESS Terminology

An "activity " is the item being scheduled. It may be anything from a complex scientific experiment to a single use of a communications link. An activity may thus consist of more than one event, with each event represented as one envelope. An activity is represented to PRESS in the form of one or more "resource envelopes".

Each "resource envelope" is equivalent to an event and represents a time period, one or more resources whose use is required, and a usage level (if appropriate). It is assumed (for scheduling purposes) that, within an envelope, resource usage is stable. Since "resource envelope" represents the requests for scheduling that will come to PRESS, we will refer to them as "request envelopes", in order to avoid confusion with the representation of the resources themselves.

Examples of "resources" are power, an instrument, a communications link, etc.

The "schedule" or "schedule timeline" is produced by the system to show what activities have been scheduled within a given time period. PRESS output shows the activities plotted against the resource used over time, so that the schedule timeline is actually a set of parallel timelines, one for each resource, with usage periods identified with an activity identifier.

VI. PRESS System Description

PRESS will accept user input interactively or from a stored file, and will be capable of creating a new schedule or modifying an existing schedule. PRESS will generate as its final product a schedule timeline, available both graphically and in a printed report. Some of the PRESS functions, described below have yet to be implemented; others exist in the current form of the rapid prototype. The discussion corresponds to the "and-or" graph of PRESS functions shown in Figure 2, with some key issues expanded in section VII.

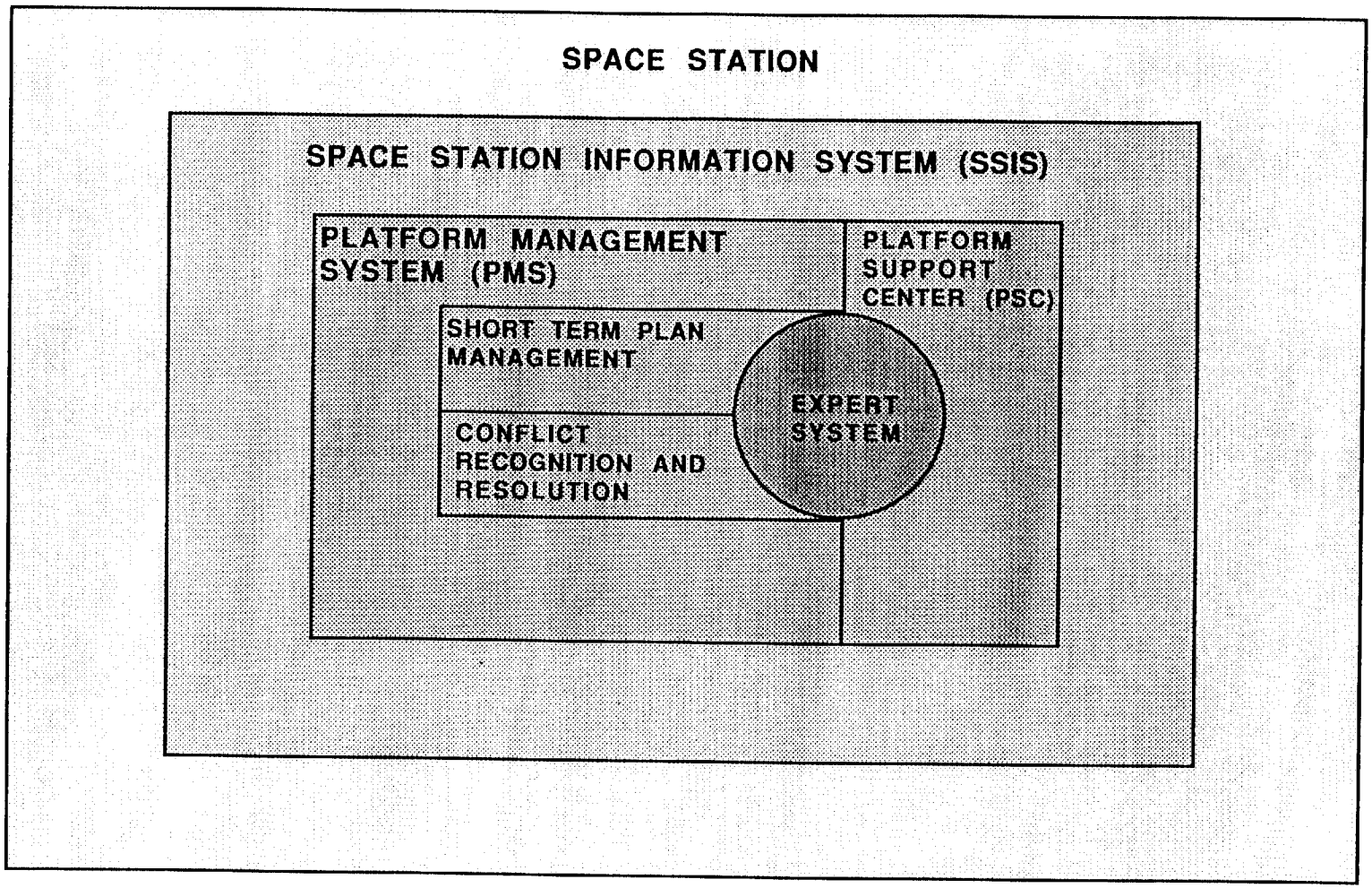


Figure 1. Placement of Expert System in PMS

FUNCTIONS OF FINAL PROTOTYPE	RAPID PROTOTYPE
perform initial planning	yes
-- process requests by priorities	yes
perform replanning	adds only
accept schedule modification requests as resource envelopes	yes
accept schedule modification requests as resource availability changes	no
resolve schedule conflicts based on assigned envelope priorities	no
perform event conflict checking via table look-up	no
accept multiple envelope requests	no
change scope of user interaction at operator's discretion	requires interaction
-- provide advice identifying modifications to the resource envelope requests that would permit successful scheduling	yes
-- provide capability for operator to cease processing before completion	yes
perform input error checking	
-- on query responses	built-in only
-- on time format	some
generate new schedule timeline output	no
-- graphic representation	yes
-- report tables	yes
	yes

Table 1. PRESS Functions

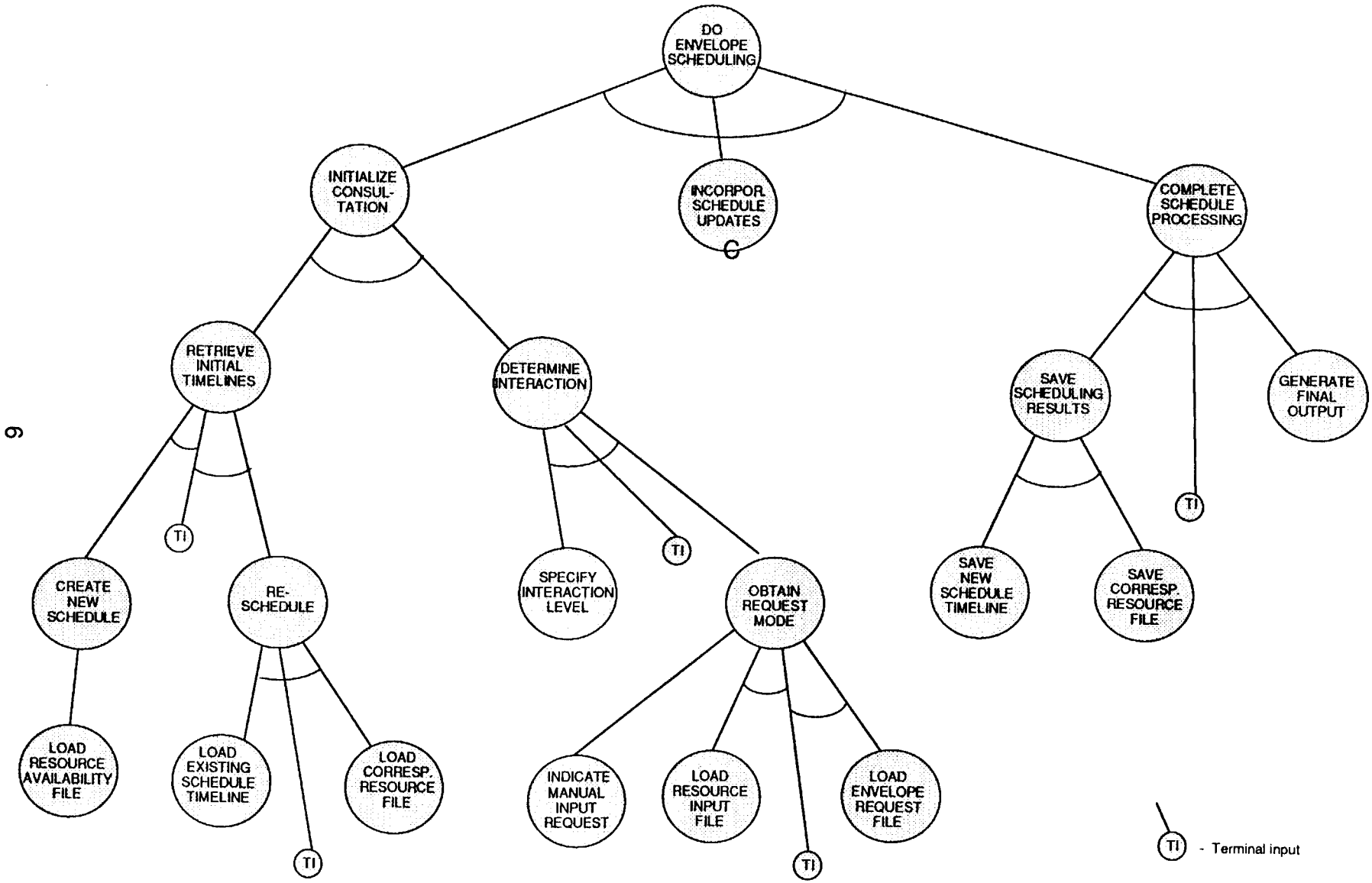


Figure 2. And-Or Graph of PRESS Functions (1 of 3)

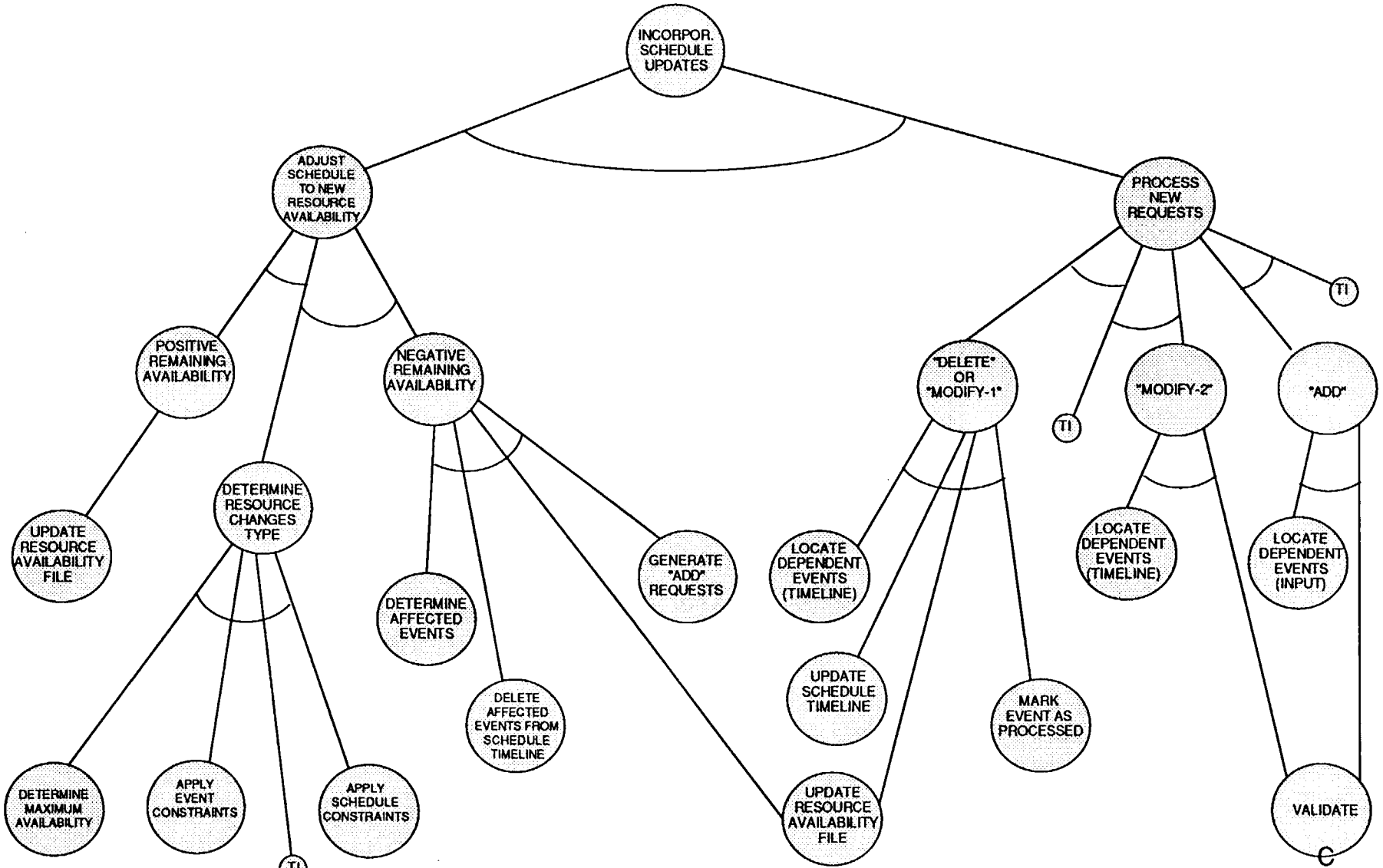


Figure 2. And-Or Graph of PRESS Functions (2 of 3)

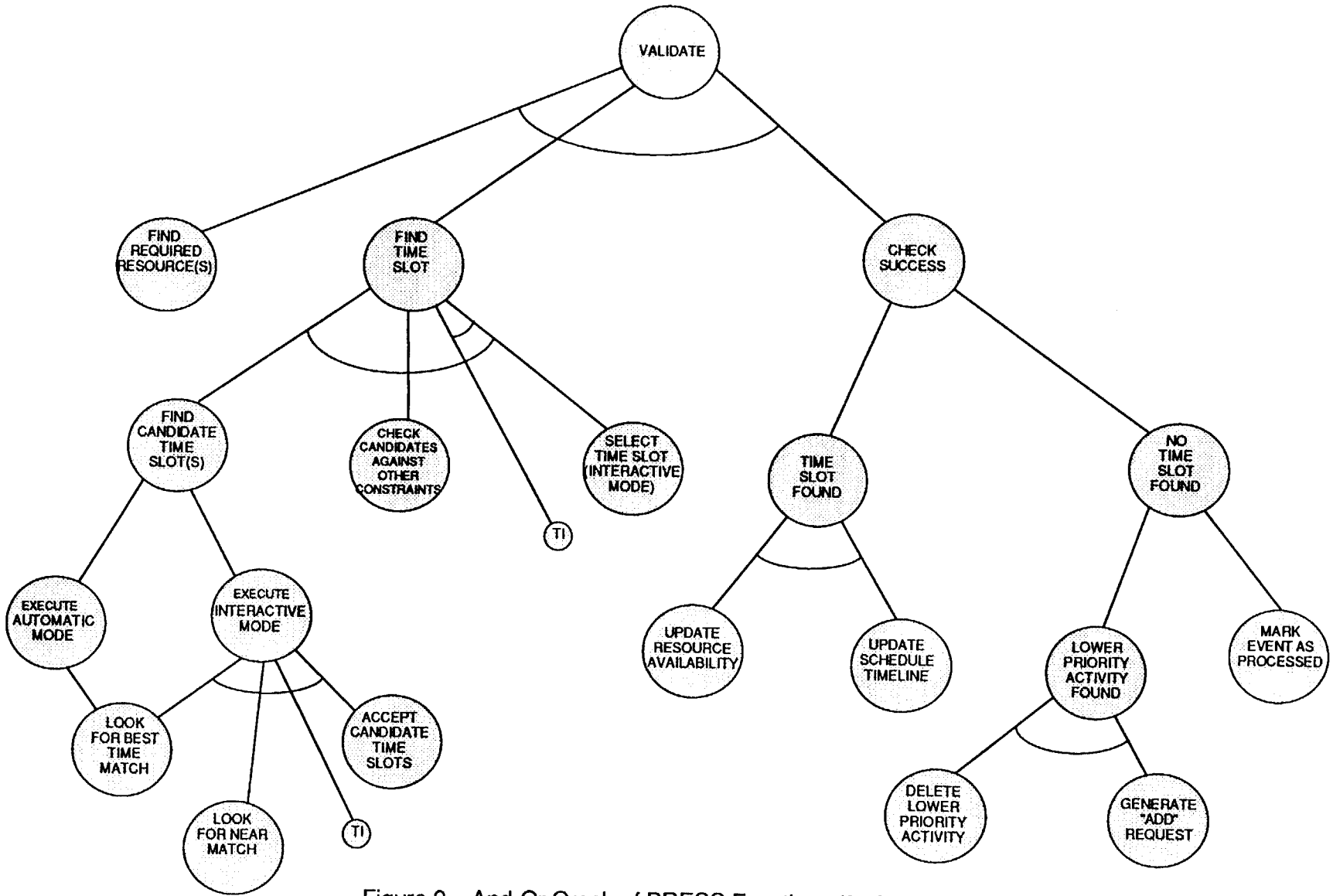


Figure 2. And-Or Graph of PRESS Functions (3 of 3)

Initialize Scheduling Consultation

Initialization will be performed once for each consultation session. The user must input information regarding the data input "mode", the location of the initial timeline, and the required level of interaction. These modes will be active through the entire consultation and cannot be changed within the same session.

PRESS will load initial files based on user-provided input including the start and end times for the requested schedule; whether the request is for the creation of a new schedule or for the replanning of an existing schedule; and the designation of the schedule immediately preceding the current schedule chronologically. For an initial planning effort, the schedule timeline will be empty, and resource availability information will be loaded from the default Resource Availability file. For replanning (modifying an existing schedule), the user-specified files, including the existing Schedule Timeline, and the corresponding Resource Availability file, will be loaded.

The user will determine the mode of system execution, including the level of operator interaction (automatic or nonautomatic) and the request mode (schedule plan/replan or change in resource availability). If the nonautomatic mode is selected, the user can specify a number of requests (N), so that, after the automatic processing of every N requests, control returns to the operator. This allows the examination or graceful termination of the system before processing of all requests is completed. Also in nonautomatic mode, an advisory feature is available which allows PRESS to widen the requested time windows to present some alternatives for an otherwise unschedulable request. This is discussed in more detail below. The user may also select manual request input, which requires the user to input request envelopes manually on a one-by-one basis. Automatic mode executes scheduling without operator interaction and without the advisory capability.

Incorporate Schedule Updates

Requests for changes in resource availability are satisfied by processing modifications to the Resource Availability file. Requests for activity scheduling or rescheduling are satisfied by processing the requests for addition, deletion, or modification of activities in the form of request envelopes. The output of this function will be the modified Schedule Timeline and the Resource Availability file.

PRESS will accept input identifying resource availability changes such as equipment failures. This type of input will include times, if applicable, and the corresponding, changed resource limit(s). Two special processing steps are required. First, the available resource level must be updated by inserting the new limits. Second, all scheduled activities drawing on the changed resource must be reexamined; they may not still be schedulable.

PRESS will compute new resource availability as a function of the following components:

$$R = F(O, N, S, C)$$

where R - new remaining resource availability (can be positive or negative)
O - currently available level of resource

N - requested changes of resource availability
S - scheduled activities drawing on the resource
C - other constraints applied to the resource.

A positive or zero value of the remaining resource availability means that the existing Schedule Timeline is still valid and to complete the consultation, it is only necessary to update the Resource Availability file. Negative value of the remaining resource availability means that the existing Schedule Timeline is no longer valid. To resolve the negative resource availability level, PRESS will identify the affected activities and delete them from the existing Schedule Timeline, creating a positive resource availability level. The changes will be reflected in the Resource Availability file, and requests to add those activities back to the schedule will be generated; they can thus be rescheduled in priority order up to the limit of the available resources.

PRESS will permit users to request the deletion or modification of a previously scheduled activity, possibly involving a series of envelopes. PRESS will automatically locate all envelopes affected by the changes and update the Schedule Timeline and the corresponding Resource Availability file. Should the modification involve an increased use of resources, PRESS will locate dependent events on the Schedule Timeline and will try to reschedule all involved events. The new resource requirements will be validated in the same way as adding a new request.

In adding a single request envelope or a series of request envelopes comprising a single activity, the system will first check the list of requests for any related envelopes. All the related envelopes will then be validated together.

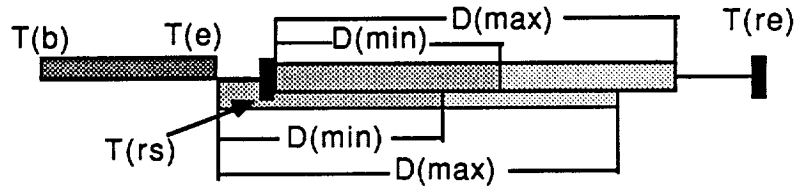
Validation

PRESS will begin the validation process by locating the requested resource(s) in the Resource Availability file. When these resources are found, PRESS will try to schedule an activity exercising the maximum duration scheduling and conflict handling concepts (these will be discussed in detail in Section VII). Once a time slot has been found, the legality of the activity at that time will be checked against other constraints, such as incompatibility with any already scheduled activities or orbital events. If no conflicts are encountered, the request is flagged as successfully scheduled. Otherwise it is passed to the conflict resolution function.

When running in fully interactive mode, PRESS will execute an advisory feature which suggests to the user, in cases where no match is found within the initial (i.e. envelope-specified) time range, what "compromises" can be made in moving the start time or shortening the duration in order to successfully schedule the request. If the user agrees to consider the suggested (altered) times, they are added to a list. When the system has exhausted possible compromises, the user is presented with the list of candidate time slots and asked to choose one or none. If the user selects none, the request is not scheduled; otherwise, the selected times are used, and the request is flagged as successfully scheduled.

Figure 3 depicts the situations where this advisory feature can be applied. In Case 1, PRESS has located an available resource within the requested duration window, but not

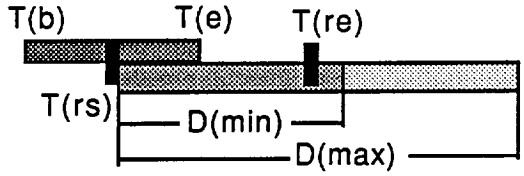
PRESS advises user to move start time to T(rs)



T(rs) >T(e); T(rs) < T(e) + D(max);
T(re) - T(rs) >= D(min)

CASE 1.

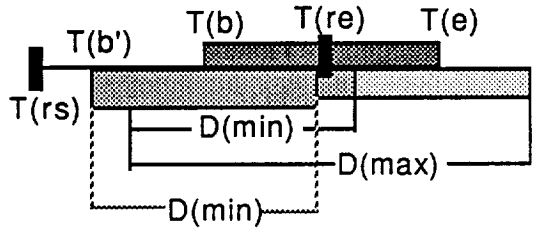
PRESS advises user to reduce duration time to T(re)-T(rs)



T(b) <= T(rs) <= T(e)
T(re) - T(rs) < D(min)

CASE 2.

PRESS advises user to move start time to T(b')

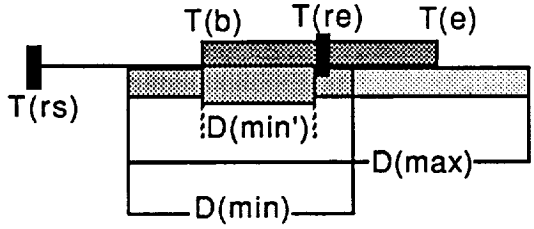


T(re)-T(b) < D(min); T(re) - D(min) = T(b')
T(rs) <= T(b') < T(b); T(b) < T(re)

CASE 3

if user does not agree

PRESS advises user to reduce duration time to D(min')



T(re)-T(b) < D(min);
T(rs) <= T(b) < T(re)

-- requested start times window
-- requested minimum duration
-- requested maximum duration
-- available resource time interval

T(b) - earliest start time
T(e) - latest start time
D(min) - minimum time duration
D(max) -maximum time duration
T(rs) - resource start time
T(re) - resource end time
T(b') - advised start time
D(min') - advised duration

Figure 3. PRESS Advisory Feature Applications

within the requested start time window. PRESS therefore computes the earliest available start time, which in this case will be the resource availablity start time, and advises the user that if the start time may be moved, then a candidate scheduling slot has been found. Case 2 illustrates a possible compromise in altering duration. Case 3 illustrates the situation in which the possibility of either altering the start time or shortening the duration may be considered.

If the scheduling attempt is successful, the updates are reflected in the Schedule Timeline and the Resource Availability file. If the attempt fails, "delete" and "add" requests will be generated for the lowest priority conflicting activity, and PRESS will choose the next request for processing. Because "delete" requests have the highest priority, the requested deletion will be processed before another attempt is made to schedule the activity. This process will iterate until either the current activity is scheduled or the current activity becomes the lowest priority activity and is deleted itself. In that case, the request is marked for no further processing and is included in the final report to the user as an unscheduled request.

The final Schedule Timeline will be saved and reported to the user at the end of the run. The user will have the option of reentering the scheduling process before exiting from PRESS. The Schedule Timeline and the Resource Availability file will be stored along with continuity information regarding how the time period for the current schedule fits with other scheduled runs. Graphics representing the Schedule Timeline will be displayed along with an option to display the resource timeline and continuity information. The user will be provided with information on any requests that could not be satisfied and will be given an opportunity to place any replanning requests for the scheduled time. A file containing a printable report will be generated at the user's request.

VII. Key Issues Addressed by PRESS

Request Envelope Types

Activity scheduling requests will be typed according to the following: "delete" a currently scheduled event; "modify" a currently scheduled event; and "add" an event not yet scheduled. "Modify" requests are divided into "modify-1", a request that involves no increase of any resource utilization and a decrease in the use of at least one resource, and "modify-2", a request involving either the same or an increased level of resource utilization.

Priority

A two-step scheme is envisioned for PRESS. Its intent is to minimize internal rescheduling/backtracking by ensuring that PRESS is always doing the most important job of which it is aware. Priority conflicts will occur primarily when a high-priority activity must be added to a previously prepared schedule.

The first criterion for determining the priority of an input request is the activity request type. Before assigning priority, PRESS must examine "modify" requests to classify them as either "modify-1" or "modify-2". "Delete" and "modify-1" requests, processed in any

order, are PRESS top-priority actions, because they return resources to the system. "Add" and "modify-2" requests are considered when there are no outstanding "delete" and "modify-1" requests. Their priorities are assumed to be supplied externally. The chosen request is marked after being processed by the system, whether successfully scheduled or not, so that it will be ignored when the next request is chosen.

Maximum Duration Scheduling

Each envelope submitted for scheduling will have an associated start time window (earliest and latest start time) and duration window (minimum and maximum duration). First, PRESS will attempt to schedule the earliest possible start time within the start time window and the maximum duration. If this attempt fails, PRESS will try to find the earliest start time and the maximum duration allowed by the resources' availability, still within the requested time window. Figure 4 illustrates possible ways in which an optimum time slot may be found.

This strategy tends to minimize backtracking by assuring that an envelope is never considered unschedulable because the wrong subset of acceptable time was chosen for the envelope. The strategy also biases scheduling in favor of high-priority activities, by assigning them the maximum amount of available time. (This scheduling strategy could be refined by giving the conflict resolution function awareness that a low priority activity might be scheduled by decreasing the duration of a conflicting high priority activity).

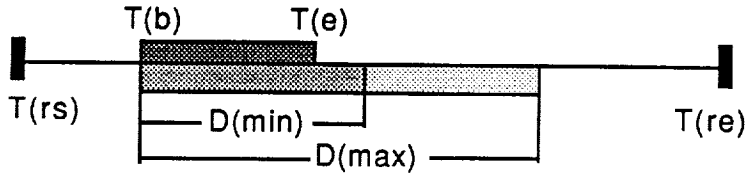
Multiple Envelopes

One of the key issues yet to be handled is the scheduling of multiple-envelope requests (i.e. activities composed of more than one event, and thus represented by more than one envelope). Scheduling a multiple-envelope activity imposes additional constraints on the system due to the need to schedule all of the activity component events, in the proper sequence. The scheduled times of the multiple-envelopes may involve mutual interdependencies (e.g., no time gaps permitted between envelopes, or specified time gaps required between envelopes). The system must be able to recognize an envelope as part of a series representing a single activity, both on input for scheduling and on making any schedule adjustments. When, after scheduling, any envelope in a series is adjusted, all other envelopes must also be reconsidered. We have not yet finalized an approach to handling this issue, but we make the following preliminary assumptions: requested time windows in a multiple envelope activity must contain no gaps, and the envelopes must be scheduled back-to-back chronologically. The approach we will try initially will involve the generation of "delete" and "add" requests for the entire series. We believe, particularly for this issue, that it is crucial to emphasize the use of frequent rapid prototyping and expert input in order to better define and refine the optimal correlation between data representation, user interface, and the PRESS knowledge base.

Constraint Checking

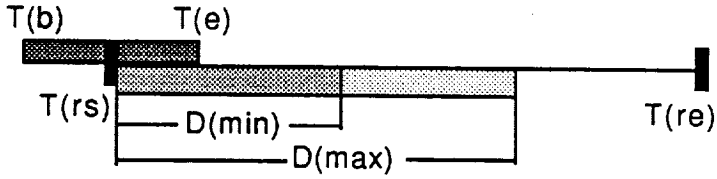
Envelopes active at the same time may conflict in one of two ways: they may

First choice found: earliest time of start window and maximum duration



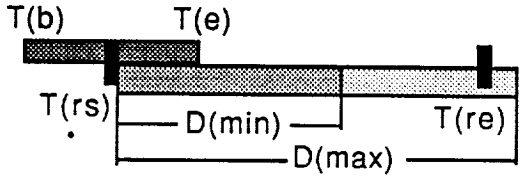
T(rs)=T(b)-n; n >=0
T(re)>=T(b)+D(max)

Acceptable time found: within requested start window and maximum duration



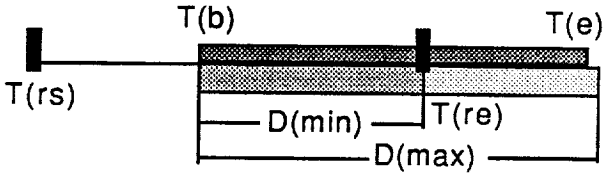
T(b) < T(rs) <= T(e)
T(re)>=T(rs)+D(max)

Acceptable time found: within requested start window and duration window



T(b) < T(rs) <= T(e)
T(rs) + D(min) <= T(re) <= T(rs)+D(max)

Acceptable time found: within requested start window and duration window



T(rs) <= T(b) < T(re)
T(re) >= T(b)+D(min)

-- requested start times window
-- requested minimum duration
-- requested maximum duration
-- available resource time interval

T(b) - earliest start time
T(e) - latest start time
D(min) - minimum time duration
D(max) -maximum time duration
T(rs) - resource start time
T(re) - resource end time

Figure 4. Optimum Time Slots with Maximum Duration Approach

oversubscribe available resources or they may require incompatible operating conditions. The first type of conflict is handled via the maximum duration scheduling approach. Operating condition constraint violations (e.g., incompatible experiments) will be checked, possibly with a combination of table lookup and rules in the knowledge base. We recognize that contextual information is critical for performance of this function.

VIII. PRESS Input and Output

PRESS will expect scheduling requests in the form of request (resource) envelopes, where each envelope represents an event, and one or more events comprise an activity. Figure 5(a) depicts the request envelope definition as it is input by the user and with the additional fields added for internal use by PRESS. The resource envelope includes a list of resource/usage-level pairs. The user will have the choice of file or manual input for scheduling requests. If the former is selected, PRESS will accept an ASCII file consisting of envelopes in list form with the required fields separated by commas. If manual input is chosen, the user must type in the envelope, when prompted by the system.

Other input required by PRESS includes a file defining the available resources and, for the replanning function, files containing the existing Schedule Timeline and the corresponding Resource Availability file.

PRESS represents each resource in list form. Figure 5(b) depicts the fields contained in a resource definition. PRESS dynamically creates and destroys resource representation lists. For an initial plan, PRESS loads a default Resource Availability file. For a replan, PRESS loads the Resource Availability file corresponding to the Schedule Timeline being considered.

PRESS outputs the result of the scheduling activity in the form of a graphic timeline and a printed report. PRESS generates the timeline dynamically, using information from an output file containing the scheduled request envelopes, annotated with the times actually scheduled. This same file is used to generate the printed report. The Resource Availability file is output containing the resource representation lists that match the scheduled activities.

IX. PRESS Development Environment

M.1 by TeKnowledge was chosen because it appeared to have many of the capabilities desired at the start of this task. These include the capability of easy interface with a language outside of the system domain, in this case, C; both forward and backward chaining capability; flexible control structures; provision for use of meta-knowledge; data representation which include a list structure; and built-in explanation facilities. In addition, the system is already into its third release and is well supported.

We are unable to provide an accurate evaluation of the M.1 shell at this point in development, but a full evaluation of the shell for the PRESS application will be part of our final report.

The M.1 software runs on an IBM PC/AT with a Video 7 EGA graphics board and an NEC Multisync color monitor. The operating system is IBM PC DOS, version 3.10.

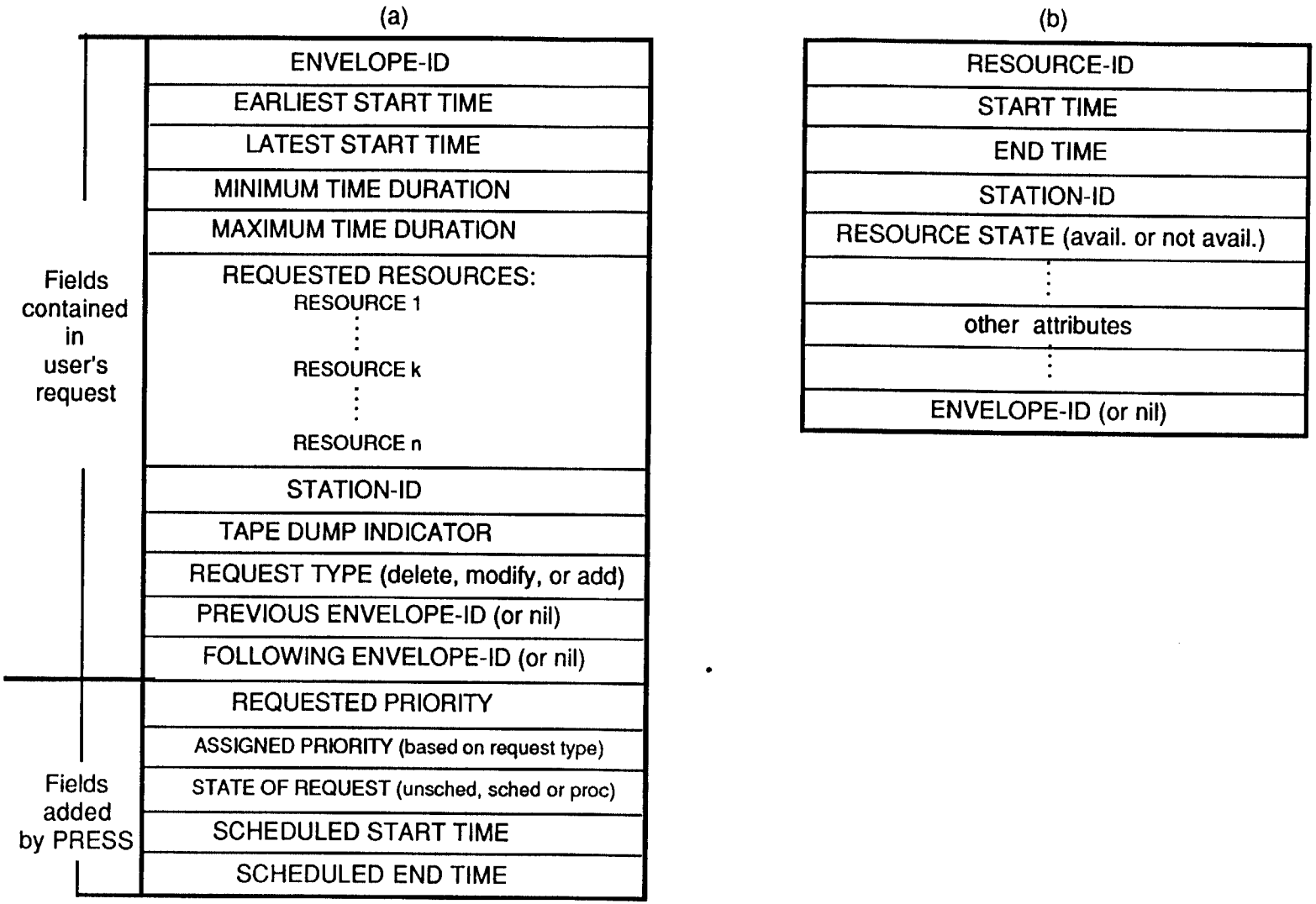


Figure 5.
Request Envelope and Resource Availability Representation

X. Comments

Several areas requiring particular attention have come up during the course of this project. The question of the nature of the user interface remains, especially in terms of the most convenient and useful input and output. Some new information on this issue is expected at the time of the rapid prototype demonstration, but the most informed feedback will arise only when PRESS is tested by operators in an approximate real-life situation. Another side of the user interface question is the level of human interaction required. Since PMS aims include the eventual migration of software from ground to on-board, we are attempting to give this issue some attention. Our initial response has been to make the system as flexible as possible by allowing the level of human interaction to be specified at the start of a consultation. Currently, if the fully automatic mode is chosen, PRESS will be unable to extend any additional flexibility in time-slot scheduling beyond that provided in the original request envelope. PRESS may help to identify types of requests to which automatic scheduling option may be applied. This function requires additional knowledge allowing the system to choose, on its own, from a list of candidate schedule opportunities.

Two of the most difficult technical issues to be solved by PRESS are multiple-envelope handling and constraint checking. These issues have been only superficially discussed here because our approach to solving them has not yet been fully defined. Both of these areas will require further definition by area experts, and it is hoped that the PRESS prototype will help to elicit that information.

Acknowledgements

The authors wish to acknowledge Ed Lewis (GSFC code 520), Dolly Perkins (GSFC code 520), Steve Tompkins (GSFC code 511), Steve Wadding (GSFC code 511), and Larry Zeigenfuss (GSFC code 511) for their encouragement and helpful discussions. This work was funded by the National Aeronautics and Space Administration, Goddard Space Flight Center for the Mission Operations Division (GSFC code 510) and the Data Systems Technology Division (GSFC code 520) under task assignment 319000.

References

"The Platform Management System Definition Document", GSFC, October, 1986
"Expert System Feasibility Study Report", Computer Sciences Corporation, September, 1986
"The PRESS Functional Definition" (initial input), Computer Sciences Corporation, February, 1987

Section B

Fault Isolation/Diagnosis Expert Systems